# File Aggregation in Enstore (Small Files)

https://cdcvs.fnal.gov/redmine/projects/show/fileaggregation

**Alexander Moibenko**

# Problem

- Writing or reading a tape mark (EOF) at the end of a file takes about 3 seconds. Writing or reading a full tape of continuous data takes just under 2 hours at top speed.
  - Thus, a tape full of 360 MB files would take twice as long —4 hours.
  - So files ought to be much larger; a few GB is good.
- And as tape capacity and speeds grow, the minimum desirable file size increases also. “Eventually, any file becomes small.”

# Project Goals

- Automatically aggregate small files into larger “container” files, with configurable definitions of “small” and “larger.”
- Transparently aggregate user's files through existing enstore interface (encp)
- Assume custodial ownership while staged to disk awaiting aggregation
- Preserve end-to-end check-summing
- Per customer "small file" policies

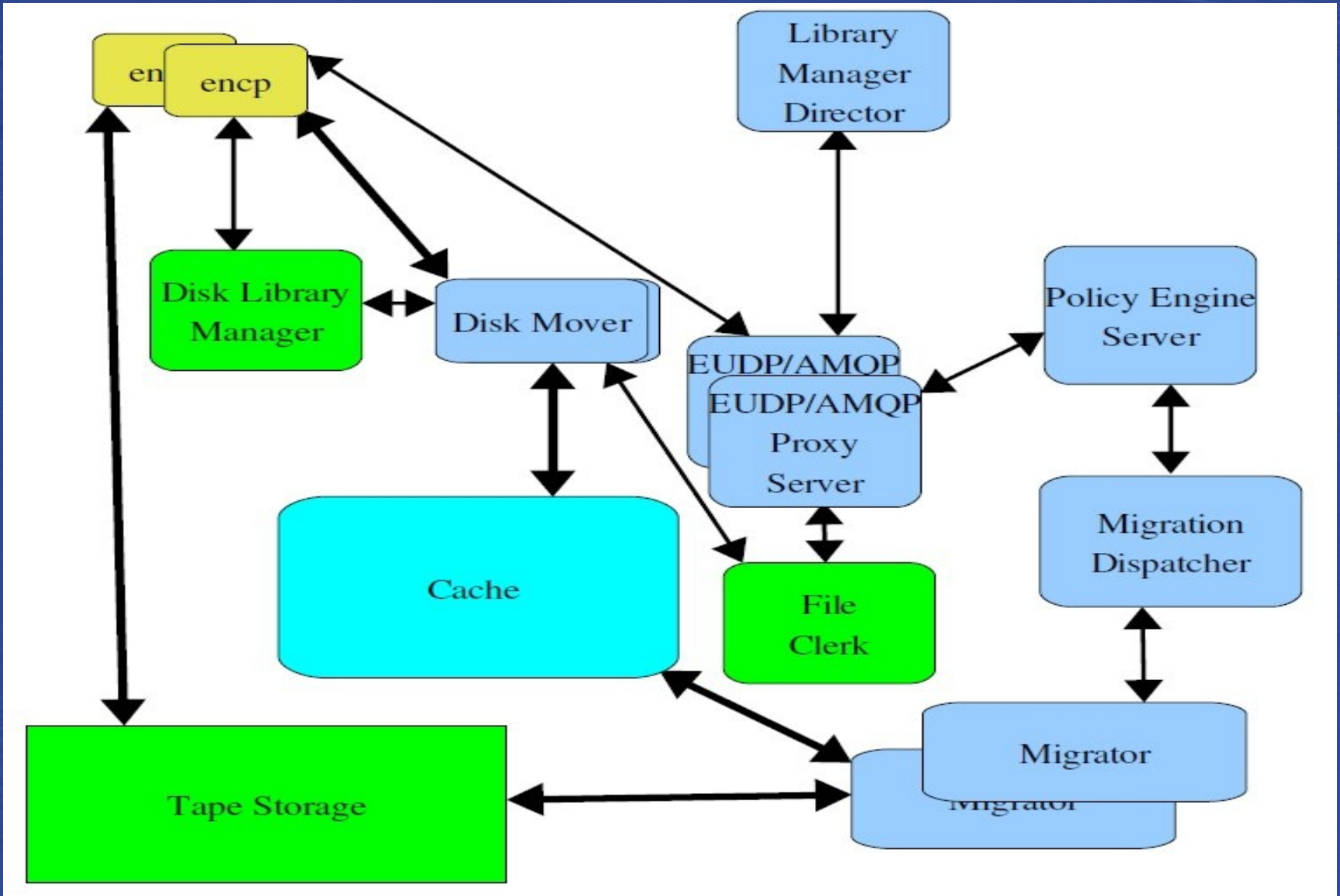
# What files to aggregate ?

- Aggregation of files shall account for read access patterns. Only the experiment, or no one, knows what read patterns will be.
- File aggregation policy must be flexible enough to adapt to different patterns without changing code.
- Aggregation of files by file family and directory trees is good to start with.

# Implementation requirements

- **File caching component** inside Enstore
  - temporary storage for incoming files, containers, and unpacked files – must be almost as safe as tape.
  - full control over cache disk access to optimize
    - IO bandwidth to tape
    - concurrent Read and Write operations
  - enstore disk movers can access any unpacked file in cache; tape movers can access any container.
- Compatibility with name-space (pnfs now, chimera later) and current client tools (encp).

# Structure of integrated data caching and tape system using encp and disk movers.



# Description of components

- Library Manager Director – receives write requests from encp and determines whether to send data to tape directly or to cache for aggregation.
- Disk Library Manager – modified enstore library manager configured for disk movers
- Disk Movers – modified enstore disk movers
- File Clerk – modified file clerk, sends events to Policy Engine Server
- EUDP2AMQP proxy server – translates messages from enstore UDP protocol to AMQP and back.

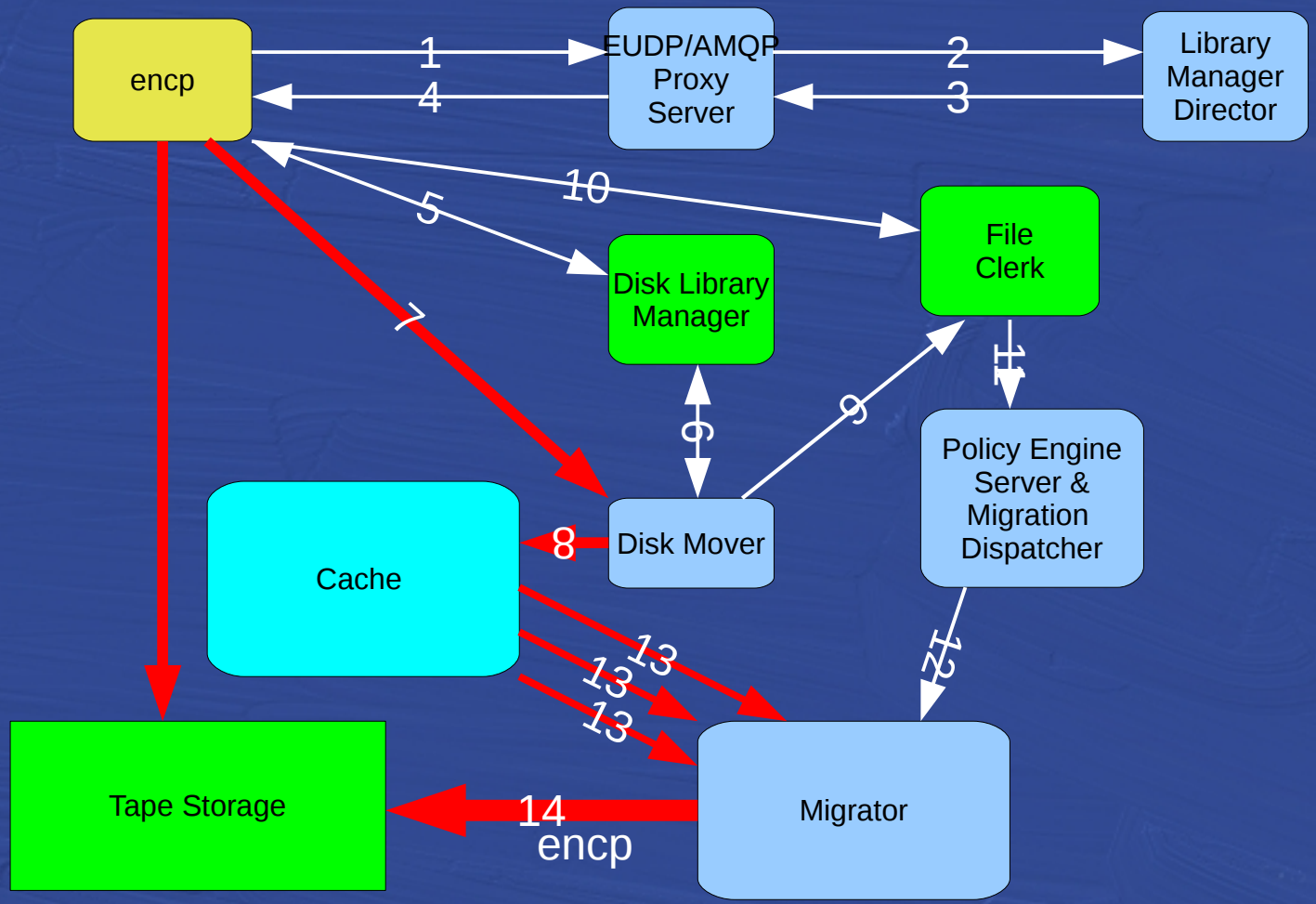
# Description of components (cont.)

- Policy Engine Server receives event from file clerk specifying that the **new** file arrived into cache or the file written to tape needs to get staged from it. PE Server has 3 types of file lists:
  - Archive – files to be written to tape.
  - Stage – files to be staged from tape(s) to cache.
  - Purge – files to be purged in cache.
- Migration dispatcher – receives file lists from Policy Engine Server and dispatches them to migrators
- Migrators aggregate data in cache and write containers to tape. They also stage aggregated data and unpack files for read requests. All files in a container read from tape get unpackaged and cached, even if not requested.

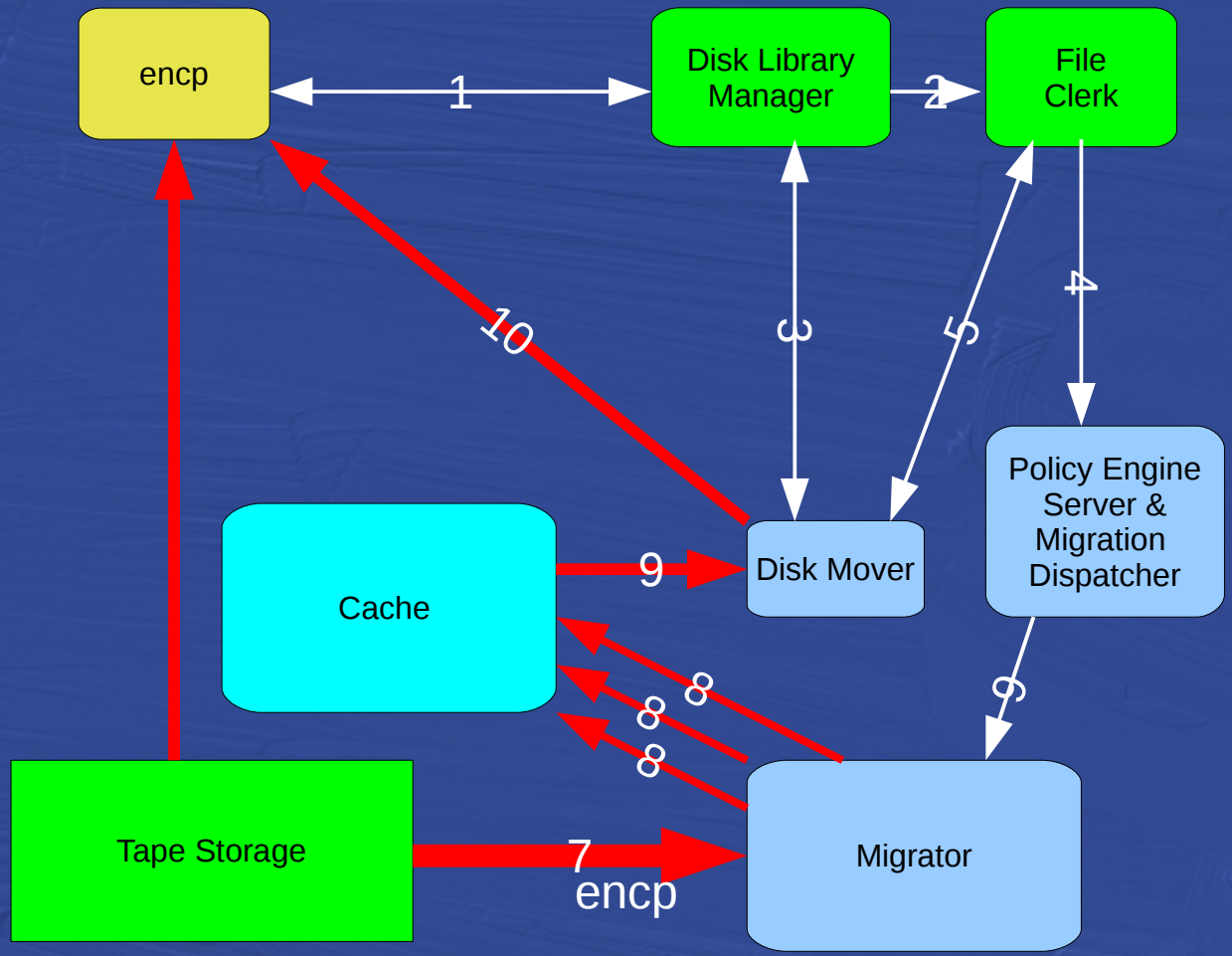
# New enstore servers

- New enstore servers are
  - Udp2amqp proxy server
  - LM director
  - Dispatcher (Policy Engine Server & Migration Dispatcher)
  - Migrator
- Currently all servers are implemented in python, later some of them, requiring  use of Policy engine will be implemented in Java (LM director, Dispatcher).

# Writing files.



# Reading files.



# Installation

- New rpm “enstore_cache....rpm” to not iterfere with installed enstore rpm
- enstore_cache rpm
  - installs code into /opt/enstore_cache
  - Modifies /etc/sudoers to run migartors
  - Creates /etc/init.d/qpid_broker to automatically start qpid_broker

# New configuration entries

AMQP broker (not enstore server):

```
configdict['amqp_broker'] = {
    'host':enstore_qpid_broker_host,
    'port':5672,
 }
```

# New configuration entries (cont.)

Udp2amqp proxy server (multiple):

```
configdict['lmd.udp_proxy_server'] = { # udp to amq proxy server
   'norestart':'INQ',
   'host': 'dmsen02.fnal.gov',
    'port': 5601,
    'udp_port' : 7710,  # udp messages come through this port
    'target_addr':'udp_relay_test', # amqp queue name
    }
```

# New configuration entries (cont.)

Library Manager Director (unique):

```
configdict['lm_director'] = {
    'norestart':'INQ',
    'host': 'dmsen02.fnal.gov',
    'port': 5602,
    'udp_port' :5602,
    'logname':'LMDSRV',
    'queue_in': 'udp_relay_test',
    'udp_proxy_server': 'lmd.udp_proxy_server',
    'policy_file': "/home/enstore/policy_files/lmd_policy.py"
    }
```

# New configuration entries (cont.)

## Dispatcher (unique):

```
configdict['dispatcher'] = {
    'host': 'dmsen02.fnal.gov',
    'port': 5603,
    'logname':'DISP',
    'queue_work': 'policy_engine', # input queue name for events (from file clerk)
    'queue_reply': 'file_clerk',
    'migrator_work': 'migrator', # common migrators input queue name
    'migrator_reply': 'migrator_reply', # common migrators reply queue name
    'policy_file': "/home/enstore/policy_files/lmd_policy.py",
    'max_time_in_cache': 600, # consider to purge files in cache if longer than this time
    'purge_watermarks':(.8, .4), # start purging if occupied > .8, stop if .4
    }
```

# New configuration entries (cont.)

## Migrator (multiple):

```
cconfigdict['M1.migrator'] = {
    'host':'dmsen02.fnal.gov',
    'port': 5610,
    'logmname': "M1MG",
    'data_area': write_cache_area, # put written files here
    'archive_area': archive_area,   # use this area for archiving written files
    'stage_area': stage_area, # use this area to keep staged from tape files
    'tmp_stage_area': tmp_stage_area, # tepmorary area for staging files from tape
    'norestart':'INQ',
    'migration_dispatcher':'dispatcher' # my migration dispatcher
    }
```

# Enstore commands for new servers

Udp2amqp proxy server:

```
[enstore@dmsen02 test_dir]$ enstore udp
Usage:
      udp [ -ha --alive --help --retries= --timeout= --usage ] udp_proxy_server
  -a, --alive              prints message if the server is up or down.
  -h, --help               prints this message
      --retries <ALIVE_RETRIES>  number of attempts to resend alive requests
      --timeout <SECONDS>   number of seconds to wait for alive response
      --usage              prints short help message
```

# Enstore commands for new servers (cont.)

## Library Manager Director:

```
[enstore@dmsen02 test_dir]$ enstore lmd
Usage:   lmd [OPTIONS]...
   -a, --alive              prints message if the server is up or down.
       --do-alarm <DO_ALARM>  turns on more alarms
(snip ...)
       --load               load a new policy file
       --retries <ALIVE_RETRIES>  number of attempts to resend alive requests
       --show               print the current policy in python format
       --timeout <SECONDS>   number of seconds to wait for alive response
       --usage              prints short help message
```

# Enstore commands for new servers (cont.)

## Dispatcher:

```
[enstore@dmsen02 test_dir]$ enstore disp
Usage:       disp [OPTIONS]...
  -a, --alive              prints message if the server is up or down.
      --do-alarm <DO_ALARM>  turns on more alarms
(snip ...)
      --get-queue          print content of pools
      --load               load a new policy file
      --retries <ALIVE_RETRIES>  number of attempts to resend alive requests
      --show               print the current policy in python format
(snip ..)
```

# Enstore commands for new servers (cont.)

## Dispatcher get-queue command:

```
[enstore@dmsen02 test_dir]$ enstore disp --get-q
{'cache_missed': {},
 'cache_purge': {},
 'cache_written': {},
 'migrartion_pool': {}}
```

# Enstore commands for new servers (cont.)

## Migrator

[[enstore@dmsen02 test_dir]$ enstore mig

Usage:

mig [ -ha --alive --help --retries= --timeout= --usage ] migrator

-a, --alive prints message if the server is up or down.

-h, --help prints this message

--retries <ALIVE_RETRIES> number of attempts to resend alive requests

--timeout <SECONDS> number of seconds to wait for alive response

--usage prints short help message

# File clerk command line options

--list – shows files inside of package, location cookies for all files in the package are the same

--package --list  -- shows all non-packaged files and package files on a tape

--pkginfo --list  -- for packaged files show additional information

--replay – re-send all events in files_in_transiton table

# Encp option

--enable-redirection – for write requests communicate with Library Manager Director for the “correct” Library Manager based om the policy for a given (from pnfs tag) Library Manager

# Monitoring

- Enstore alarms
- Enstore web pages
  - http://dmsen02.fnal.gov/enstore/enstore_saag.html
  - http://dmsen02.fnal.gov/enstore/status_enstore_system.html
- Web based qpid broker monitor.
- Monitoring and audition scripts under development.